Inf2D 12: Resolution-Based Inference

Cristina Alexandru

School of Informatics, University of Edinburgh

09/02/18

Slide Credits: Jacques Fleuriot, Michael Rovatsos, Michael Herrmann
Last time

- Unification: Given $\alpha$ and $\beta$, find $\theta$ such that $\alpha\theta = \beta\theta$
- Most general unifier (MGU)

- Generalised Modus Ponens

\[
p_1', p_2', \ldots, p_n' (p_1 \land p_2 \land \cdots \land p_n \Rightarrow q) \quad \text{when} \quad p_i'\theta \equiv p_i\theta \quad \forall i
\]
Outline

- Forward chaining
- Backward chaining
- Resolution
Forward chaining algorithm

```
function FOL-FC-Ask(KB, α) returns a substitution or false
    inputs: KB, the knowledge base, a set of first-order definite clauses
            α, the query, an atomic sentence
    local variables: new, the new sentences inferred on each iteration

    repeat until new is empty
        new ← {}
        for each rule in KB do
            (p₁ ∧ ... ∧ pₙ ⇒ q) ← STANDARDIZE-VARIABLES(rule)
            for each θ such that SUBST(θ, p₁ ∧ ... ∧ pₙ) = SUBST(θ, p'₁ ∧ ... ∧ p'ₙ)
                for some p'₁, ..., p'ₙ in KB
                    q' ← SUBST(θ, q)
                    if q' does not unify with some sentence already in KB or new then
                        add q' to new
                        φ ← UNIFY(q', α)
                        if φ is not fail then return φ
                add new to KB
        return false
```
American\( (x) \) \& Weapon\( (y) \) \& Sells\( (x, y, z) \) \& Hostile\( (z) \) \\rightarrow Criminal\( (x) \)

Owns\( (Nono, M_1) \) and Missile\( (M_1) \)

Missile\( (x) \) \& Owns\( (Nono, x) \) \\rightarrow Sells\( (West, x, Nono) \)

Missile\( (x) \) \\rightarrow Weapon\( (x) \)

Enemy\( (x, America) \) \\rightarrow Hostile\( (x) \)

American\( (West) \) and Enemy\( (Nono, America) \)
Forward chaining proof

\[
\begin{align*}
\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) & \Rightarrow \text{Criminal}(x) \\
\text{Owns}(\text{Nono}, M_1) \text{ and } \text{Missile}(M_1) & \\
\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) & \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \\
\text{Missile}(x) & \Rightarrow \text{Weapon}(x) \\
\text{Enemy}(x, \text{America}) & \Rightarrow \text{Hostile}(x) \\
\text{American}(\text{West}) \text{ and } \text{Enemy}(\text{Nono, America}) & 
\end{align*}
\]
Forward chaining proof

\[
\begin{align*}
\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) & \Rightarrow \text{Criminal}(x) \\
\text{Owns}(\text{Nono}, M_1) \text{ and } \text{Missile}(M_1) & \\
\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) & \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \\
\text{Missile}(x) & \Rightarrow \text{Weapon}(x) \\
\text{Enemy}(x, \text{America}) & \Rightarrow \text{Hostile}(x) \\
\text{American}(\text{West}) \text{ and } \text{Enemy}(\text{Nono, America}) & 
\end{align*}
\]
Forward chaining proof

\[
\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)
\]
\[
\text{Owns}(\text{Nono}, M_1) \text{ and } \text{Missile}(M_1)
\]
\[
\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})
\]
\[
\text{Missile}(x) \Rightarrow \text{Weapon}(x)
\]
\[
\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)
\]
\[
\text{American}(\text{West}) \text{ and } \text{Enemy}(\text{Nono, America})
\]
Sound and complete for first-order definite clauses
- Definite clause = exactly one positive literal.

**Datalog** = first-order definite clauses + no functions
- FC terminates for Datalog in finite number of iterations

May not terminate in general if \( \alpha \) is not entailed
This is unavoidable: entailment with definite clauses is semi-decidable
Incremental forward chaining: no need to match a rule on iteration $k$ if a premise wasn’t added on iteration $k - 1$ \Rightarrow match each rule whose premise contains a newly added positive literal

Matching itself can be expensive:
Database indexing allows $O(1)$ retrieval of known facts
  e.g., query $\text{Missile}(x)$ retrieves $\text{Missile}(M1)$

Forward chaining is widely used in deductive databases
Pattern Matching

For each $\theta$ such that 
$$\text{SUBST}(\theta, p_1 \land \cdots \land p_n) = \text{SUBST}(\theta, p'_1 \land \cdots \land p'_n)$$
for some $p'_1, \ldots, p'_n$ in KB

Finding all possible unifiers can be very expensive

Example

$\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$

Can find each object owned by Nono in constant time and then check if it is a missile

But what if Nono owns many objects but very few missiles?

Conjunct Ordering: Better (cost-wise) to find all missiles first and then check whether they are owned by Nono

Optimal ordering is NP-hard. Heuristics available: e.g. MRV from CSP if each conjunct is viewed as a constraint on its vars
Every finite domain CSP can be expressed as a single definite clause + ground facts

Colourable is inferred iff the CSP has a solution

CSPs include 3SAT as a special case, hence matching is NP-hard
Backward chaining algorithm

function FOL-BC-ASK(KB, query) returns a generator of substitutions
return FOL-BC-OR(KB, query, {})

generator FOL-BC-OR(KB, goal, θ) yields a substitution
for each rule (lhs ⇒ rhs) in FETCH-RULES-FOR-GOAL(KB, goal) do
(lhs, rhs) ← STANDARDIZE-VARIABLES((lhs, rhs))
for each θ' in FOL-BC-AND(KB, lhs, UNIFY(rhs, goal, θ)) do
yield θ'

generator FOL-BC-AND(KB, goals, θ) yields a substitution
if θ = failure then return
else if LENGTH(goals) = 0 then yield θ
else do
first, rest ← FIRST(goals), REST(goals)
for each θ' in FOL-BC-OR(KB, SUBST(θ, first), θ) do
for each θ'' in FOL-BC-AND(KB, rest, θ') do
yield θ''

\[ \text{SUBST(COMPOSE}(\theta_1, \theta_2), p) = \text{SUBST}(\theta_2, \text{SUBST}(\theta_1, p)) \]
Backward chaining example

$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$
$Owns(Nono, M_1) \land Missile(M_1)$
$Missile(x) \land Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
$Missile(x) \Rightarrow Weapon(x)$
$Enemy(x, America) \Rightarrow Hostile(x)$
$American(West) \land Enemy(Nono, America)$

$Criminal(West)$
Backward chaining example

\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]

\[ \text{Owns}(\text{Nono}, M_1) \text{ and } \text{Missile}(M_1) \]

\[ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]

\[ \text{Missile}(x) \Rightarrow \text{Weapon}(x) \]

\[ \text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x) \]

\[ \text{American}(\text{West}) \text{ and } \text{Enemy}(\text{Nono}, \text{America}) \]
Backward chaining example

\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]
\[ \text{Owns}(\text{Nono}, M_1) \land \text{Missile}(M_1) \]
\[ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]
\[ \text{Missile}(x) \Rightarrow \text{Weapon}(x) \]
\[ \text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x) \]
\[ \text{American}(\text{West}) \text{ and } \text{Enemy}(\text{Nono}, \text{America}) \]
Backward chaining example

\[\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)\]
\[\text{Owns}(\text{Nono}, M_1) \text{ and } \text{Missile}(M_1)\]
\[\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})\]
\[\text{Missile}(x) \Rightarrow \text{Weapon}(x)\]
\[\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)\]
\[\text{American}(\text{West}) \text{ and } \text{Enemy}(\text{Nono}, \text{America})\]
Backward chaining example

\[\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)\]

\[\text{Owns}(\text{Nono}, M_1) \land \text{Missile}(M_1)\]

\[\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})\]

\[\text{Missile}(x) \Rightarrow \text{Weapon}(x)\]

\[\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)\]

\[\text{American}(\text{West}) \land \text{Enemy}(\text{Nono}, \text{America})\]
Backward chaining example

\[
\begin{align*}
\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) & \Rightarrow \text{Criminal}(x) \\
\text{Owns}(\text{Nono}, M_1) \text{ and } \text{Missile}(M_1) & \\
\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) & \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \\
\text{Missile}(x) & \Rightarrow \text{Weapon}(x) \\
\text{Enemy}(x, \text{America}) & \Rightarrow \text{Hostile}(x) \\
\text{American}(\text{West}) \text{ and } \text{Enemy}(\text{Nono}, \text{America}) &
\end{align*}
\]
Backward chaining example

\[
\begin{align*}
\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) & \Rightarrow \text{Criminal}(x) \\
\text{Owns}(\text{Nono}, M_1) \land \text{Missile}(M_1) & \\
\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) & \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \\
\text{Missile}(x) & \Rightarrow \text{Weapon}(x) \\
\text{Enemy}(x, \text{America}) & \Rightarrow \text{Hostile}(x) \\
\text{American}(\text{West}) \land \text{Enemy}(\text{Nono, America}) &
\end{align*}
\]
Properties of backward chaining

- Depth-first recursive proof search: space is linear in size of proof
- Incomplete due to infinite loops
  - partial fix by checking current goal against every goal on stack
- Inefficient due to repeated subgoals (both success and failure)
  - fix using caching of previous results (extra space)
- Widely used for logic programming
A method for telling whether a propositional formula is satisfiable and for proving that a first-order formula is unsatisfiable.

Yields a complete inference algorithm

If a set of clauses is unsatisfiable, then the resolution closure of those clauses contains the empty clause (propositional logic)
Ground Binary Resolution

\[ \frac{C \lor P \quad D \lor \neg P}{C \lor D} \]

Soundness:

\[ C \lor P \text{ iff } \neg C \Rightarrow P \]
\[ D \lor \neg P \text{ iff } P \Rightarrow D \]

Therefore, \( \neg C \Rightarrow D \),

which is equivalent to \( C \lor D \)

Note: if both \( C \) and \( D \) are empty then resolution deduces the empty clause, i.e. false.
Non-Ground Binary Resolution

\[ \frac{C \lor P \quad D \lor \neg P'}{(C \lor D) \theta} \]

where \( \theta \) is the MGU of \( P \) and \( P' \)

- The two clauses are assumed to be standardized apart so that they share no variables.
- **Soundness**: apply \( \theta \) to premises then appeal to ground binary resolution.

\[ \frac{C \theta \lor P \theta \quad D \theta \lor \neg P \theta}{C \theta \lor D \theta} \]
Example

\[ \neg \text{Rich}(x) \lor \text{Unhappy}(x) \quad \text{Rich}(Ken) \]

\[ \frac{}{\text{Unhappy}(Ken)} \]

with \( \theta = \{x/\text{Ken}\} \)
Factoring

\[
\frac{C \lor P_1 \lor \ldots \lor P_m}{(C \lor P_1) \theta}
\]

where \( \theta \) is the MGU of the \( P_i \)

**Soundness:** by universal instantiation and deletion of duplicates.
\[
C \lor P_1 \lor \ldots \lor P_m \quad D \lor \neg P_1' \lor \ldots \lor \neg P_n' \\
\frac{(C \lor D) \theta}{(C \lor D) \theta}
\]

where \( \theta \) is MGU of all \( P_i \) and \( P'_j \)

- **Soundness**: by combination of factoring and binary resolution.
- To prove \( \alpha \): apply resolution steps to \( CNF(\text{KB} \land \neg \alpha) \)
  - complete for FOL, if full resolution or binary resolution + factoring is used
Conversion to CNF

Example:

Everyone who loves all animals is loved by someone:
\[ \forall x. [\forall y. \text{Animal}(y) \Rightarrow \text{Loves}(x, y)] \Rightarrow [\exists y. \text{Loves}(y, x)] \]

1. Eliminate all biconditionals and implications
   \[ \forall x. \neg [\forall y. \neg \text{Animal}(y) \lor \text{Loves}(x, y)] \lor [\exists y. \text{Loves}(y, x)] \]

2. Move \( \neg \) inwards, use: \( \neg \forall x. p \equiv \exists x. \neg p \), \( \neg \exists x. p \equiv \forall x. \neg p \), etc.
   \[ \forall x. [\exists y. \neg (\neg \text{Animal}(y) \lor \text{Loves}(x, y))] \lor [\exists y. \text{Loves}(y, x)] \]
   \[ \forall x. [\exists y. \neg \text{Animal}(y) \land \neg \text{Loves}(x, y)] \lor [\exists y. \text{Loves}(y, x)] \]
   \[ \forall x. [\exists y. \text{Animal}(y) \land \neg \text{Loves}(x, y)] \lor [\exists y. \text{Loves}(y, x)] \]
1. **Standardize variables apart:** each quantifier should use a different one
   \[ \forall x. [\exists y. Animal(y) \land \neg Loves(x, y)] \lor [\exists z. Loves(z, x)] \]

2. **Skolemize:** a more general form of existential instantiation
   Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables
   \[ \forall x. [Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x) \]

3. **Drop universal quantifiers:**
   \[ [Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x) \]

4. **Distribute \lor over \land:**
   \[ [Animal(F(x)) \lor Loves(G(x), x)] \land [\neg Loves(x, F(x)) \lor Loves(G(x), x)] \]

---

1) No enclosing universal quantifier? Just replace with Skolem constant, i.e. a function with no argument.
'West' Clauses

\neg American(x) \lor \neg Weapon(y) \lor \neg Sells(x, y, z) \lor
\neg Hostile(z) \lor \text{Criminal}(x)

\text{Owns}(\text{Nono}, M_1) \text{ and } \text{Missile}(M_1)

\neg \text{Missile}(x) \lor \neg \text{Owns}(\text{Nono}, x) \lor \text{Sells}(\text{West}, x, \text{Nono})

\neg \text{Missile}(x) \lor \text{Weapon}(x)

\neg \text{Enemy}(x, \text{America}) \lor \text{Hostile}(x)

\text{American}(\text{West}) \text{ and } \text{Enemy}(\text{Nono}, \text{America})
Resolution proof: definite clauses

\[ \neg \text{American}(x) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(x,y,z) \lor \neg \text{Hostile}(z) \lor \text{Criminal}(x) \]

\[ \neg \text{Criminal}(\text{West}) \]

\[ \neg \text{American}(\text{West}) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z) \]

\[ \text{American}(\text{West}) \]

\[ \neg \text{Missile}(x) \lor \text{Weapon}(x) \]

\[ \neg \text{Weapon}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z) \]

\[ \text{Missile}(\text{M1}) \]

\[ \neg \text{Missile}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z) \]

\[ \neg \text{Missile}(x) \lor \neg \text{Owns}(\text{Nono},x) \lor \text{Sells}(\text{West},x,\text{Nono}) \]

\[ \neg \text{Sells}(\text{West},\text{M1},z) \lor \neg \text{Hostile}(z) \]

\[ \text{Missile}(\text{M1}) \]

\[ \neg \text{Missile}(\text{M1}) \lor \neg \text{Owns}(\text{Nono},\text{M1}) \lor \neg \text{Hostile}(\text{Nono}) \]

\[ \text{Missile}(\text{M1}) \]

\[ \neg \text{Owns}(\text{Nono},\text{M1}) \lor \neg \text{Hostile}(\text{Nono}) \]

\[ \text{Owns}(\text{Nono},\text{M1}) \]

\[ \neg \text{Owns}(\text{Nono},\text{M1}) \lor \neg \text{Hostile}(\text{Nono}) \]

\[ \neg \text{Enemy}(x,\text{America}) \lor \text{Hostile}(x) \]

\[ \text{Enemy}(\text{Nono, America}) \]

\[ \neg \text{Hostile}(\text{Nono}) \]

\[ \text{Enemy}(\text{Nono, America}) \]
Summary

- Forward chaining
- Backward chaining
- Resolution